Министерство образования Московской области

Государственное бюджетное образовательное учреждение

Московской области

«Красногорский колледж»

Задание по Учебной практике

По специальности 09.02.07 Информационные системы и программирование

от студента

Юлдашева Иброхима Камалидиновича

Проверил: преподаватель

Лобанов Глеб Ильич

# Содержание отчёта по учебной практике

Оглавление

[Содержание отчёта по учебной практике 2](#_Toc168565353)

[Описание объекта исследования 3](#_Toc168565354)

[Введение 3](#_Toc168565355)

[Логическое моделирование. 5](#_Toc168565356)

[Логический словарь данных 6](#_Toc168565357)

[Физическое моделирование 8](#_Toc168565358)

[Описание программного продукта 11](#_Toc168565359)

[Заключение 28](#_Toc168565360)

# Введение

Объектом исследования является компания "OOO МКАД АЗС".

Актуальность данной работы обусловлена большим спросом на бензин и другие виды топлива, и увеличением на них спроса.

Цель данной работы: разработка информационной системы для учёта данных по продаже Компании "OOO МКАД АЗС"

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Написание Логической модели базы данных.
2. Составление Базы данных.
3. Создание физической модели данных.
4. Разработка API приложения по созданной базе данных.
5. Демонстрация работы ИС.
6. Сдача работы преподавателю

# Описание объекта исследования

Данная работа по учебной практике состоит в том, чтобы написать приложение по теме «Разработка информационной системы для учёта данных по продаже Компании "OOO МКАД АЗС".

Компания "OOO МКАД АЗС" занимается продажей бензина на заправках (95, 92, 98, дизель, 100), газа (бутан, пропан), а так же продукты питания в пристроенных к заправкам магазинчиками (напитки, сладости, продукты быстрого приготовления, снеки).

Данная программа должна облегчить работу сотрудников по ведению учёта о покупке топлива клиентом, какой марки было топливо, и кто является поставщиком данного топлива.

Объект исследования информационной системы заправки

В качестве объекта исследования можно рассматривать процесс управления и контроля за работой автозаправочной станции (АЗС).

Основные компоненты объекта:

инфраструктура АЗС, включающая здания, сооружения, оборудование; программное обеспечение, автоматизирующее процессы управления и учёта на АЗС;

технические средства: компьютеры, серверы, терминалы, кассовые аппараты, датчики уровня топлива и т. д.;

персонал, ответственный за обслуживание клиентов, управление оборудованием и соблюдение правил безопасности;

клиенты — физические и юридические лица, пользующиеся услугами АЗС для заправки своих транспортных средств;

поставщики топлива, поставляющие топливо на АЗС для последующей продажи клиентам;

финансовые операции, связанные с оплатой услуг АЗС клиентами и расчётами с поставщиками топлива;

системы безопасности, предотвращающие кражи, пожары, аварии и другие чрезвычайные ситуации на территории АЗС;

управление запасами, включая контроль за количеством и качеством топлива на складе АЗС и планирование закупок у поставщиков.

Исследование направлено на повышение эффективности работы АЗС путём оптимизации процессов управления, учёта и контроля с использованием современных информационных технологий. Для достижения этой цели необходимо решить следующие задачи:

проанализировать существующие методы и подходы к управлению АЗС;

разработать модель информационной системы, учитывающую особенности работы конкретной АЗС;

внедрить разработанную систему на практике и оценить её эффективность.

Анализ данных о работе АЗС за определённый период времени, моделирование различных сценариев управления запасами и финансовыми операциями, а также разработка рекомендаций по улучшению работы станции на основе полученных результатов позволят выявить слабые места и предложить пути их устранения. Это приведёт к повышению эффективности работы АЗС и удовлетворённости клиентов

# Логическое моделирование.

Логическая модель базы данных будет показывать количество сущностей, атрибутов этих сущностей, связи между сущностями, проектирование сильных и зависимых сущностей.

Логическая модель данных представлена на рисунке 1.

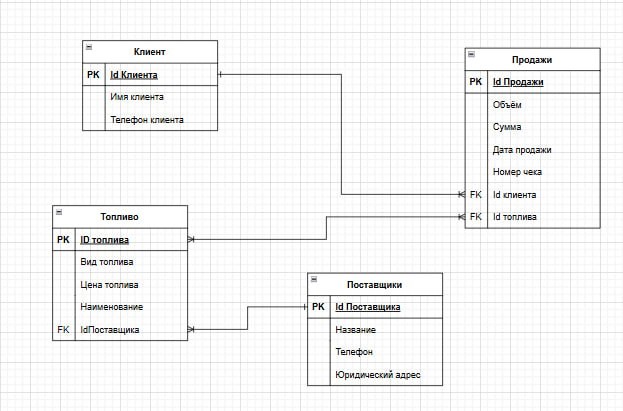


Рисунок 1. Логическая модель для базы данных на тему «Разработка информационной системы для учёта данных по продаже Компании "OOO МКАД АЗС"».

# Логический словарь данных

Логический словарь по клиенту представлен ниже

Таблица 1 Логический словарь по клиенту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Название атрибута | Тип данных | Is Null | Описание |
| Первичный ключ (PK) | ID Клиента | INT | 0 | По нему можно найти нужного клиента |
| --- | Имя клиента | STRING | 0 | Записывается информация об имени клиента |
| --- | Телефон клиента | INT | 0 | Содержится и добавляется информация о номере телефона клиента |

Логический словарь по продажам представлен ниже

Таблица 2 Логический словарь по продажам

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Название атрибута | Тип данных | Is Null | Описание |
| Первичный ключ (PK) | ID Продажи | INT | 0 | По нему можно найти нужного клиента |
| --- | Объём | STRING | 0 | записывается информация о Объёме |
| --- | Сумма | STRING | 0 | Записывается информация о Сумме покупки |
| --- | Дата продажи | DATATIME | 0 | Записывается информация о дате продажи |
| --- | Номер чека | STRING | 0 | Записывается информация о номере чека |
| Вторичный ключ (FK) | ID клиента | INT | 0 | По нему можно найти первичный ключ таблицы, от которой зависит эта таблица |
| Вторичный ключ (FK) | ID топлива | INT | 0 | По нему можно найти первичный ключ таблицы, от которой зависит эта таблица |

Логический словарь по топливу представлен ниже

Таблица 3 Логический словарь по топливу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Название атрибута | Тип данных | Is Null | Описание |
| Первичный ключ (PK) | ID топлива | INT | 0 | По нему можно найти нужного клиента |
| --- | Вид топлива | String | 0 | Записывается информация о виде топлива |
| --- | Цена топлива | String | 0 | Записывается информация о цене топлива |
| --- | Наименование | String | 0 | Записывается информация о названии топлива |
| Вторичный ключ (FK) | Поставщик ID | INT | 0 | По нему можно найти первичный ключ таблицы, от которой зависит эта таблица |

Логический словарь по поставщикам представлен ниже

Таблица 4 Логический словарь по поставщикам

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Название атрибута | Тип данных | Is Null | Описание |
| Первичный ключ (PK) | ID Поставщика | INT | 0 | По нему можно найти нужного клиента |
| --- | Наименование | String | 0 | Записывается информация о наименовании поставщика |
| --- | Телефон поставщика | String | 0 | Содержится и добавляется информация о номере телефона поставщика |
| --- | Юридический адрес | String | 0 | Записывается информация о юридическом адресе поставщика |

# Физическое моделирование

Физическая модель данных представлена на рисунке 2

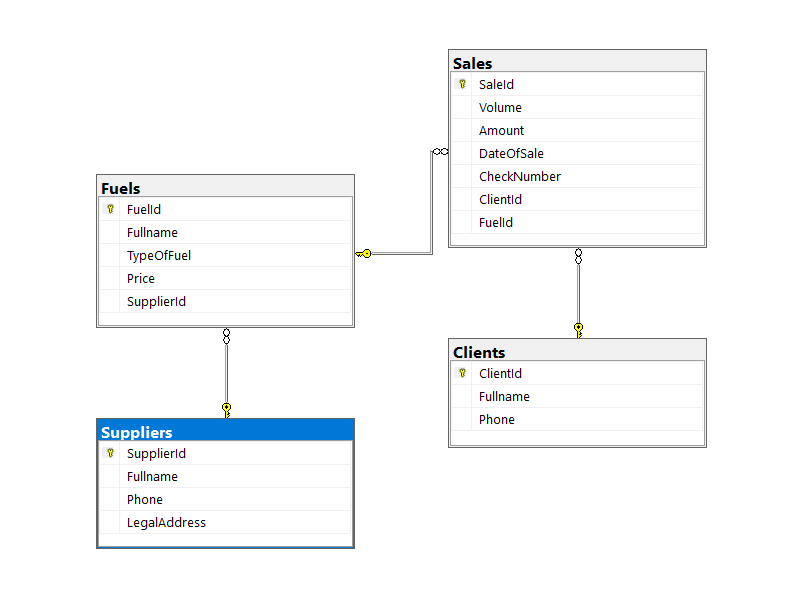


Рисунок 2 – физическая модель данных.

**Листинги базы данных представлены ниже**

Листинг запросов на создание таблиц представлен на листинге 1

Листинг 1 создание таблиц

|  |
| --- |
| Create database Autorefueling  Use Autorefueling  Create table Clients(  ClientId int identity primary key(ClientId) ,  Fullname varchar (100) not null,  Phone char(11) not null,);  Create table Suppliers(  SupplierId int identity primary key(SupplierId),  Fullname varchar(100) not null,  Phone char(11) not null,  LegalAddress varchar(100) not null  );  Create table Fuels(  FuelId int identity primary key(FuelId),  Fullname varchar(100) not null,  TypeOfFuel varchar(100) not null,  Price varchar(100) not null,  SupplierId int REFERENCES Suppliers (SupplierId)  );  Create table Sales(  SaleId int identity primary key(SaleId),  Volume varchar(100) not null,  Amount varchar(100) not null,  DateOfSale date,  CheckNumber char(11) not null,  ClientId int REFERENCES Clients (ClientId),FuelId int REFERENCES Fuels (FuelId),  ); |

Листинг запросов на заполнение таблиц представлен на листинге 2

Листинг 2 заполнение таблиц

|  |
| --- |
| insert into Clients  values  ('Иван Иванов','72343458998'),  ('Пётр Петров','79782353478'),  ('Антон Антонов','79846534256');  insert into Suppliers  values('ЮС ОИЛ-ТРЕЙД','78943653425','Ростов-на-Дону,ул.Московская, д. 53.'),  ('АВАНГАРД-ОИЛ','79202078382',' Белгород, ул. Н.Островского, д.18а.'),  ('БК ТРЕЙД','79381003628','Ростов-на-Дону, пер. Доблестный, д.8.');  insert into Fuels  values  ('АИ-92','Бензин','47р-литр','1'),  ('АИ-95','Бензин','54р-литр','2'),  ('Дизель','Дизельное топливо','37р-литр','3');  insert into Sales  values  ('100м3','2000р','2024-03-20','19283546768','1','1'),  ('123м3','1350р','2024-04-12','23453566678','2','2'),  ('125м3','22370р','2024-05-24','77756543666','3','3'); |

Листинг JOIN запросов представлен на листинге 3

Листинг 3 JOIN запросы

|  |
| --- |
| Select \* From Salesleft Join Clients  on Sales.ClientId = Clients.ClientId  Select \* From Salesleft Join Fuels  on Sales.FuelId = Fuels.FuelId  Select \* From Fuelsleft Join Suppliers  on Fuels.SupplierId = Suppliers.SupplierId  Select Fuels.Fullname as 'Название топлива',  Suppliers.Fullname as 'Название поставщика ' From Fuelsleft Join Suppliers  on Fuels.SupplierId = Suppliers.SupplierId  Select Sales.CheckNumber as 'Номер Чека', Clients.Fullname as 'Имя Клиент ' From Sales left Join Clients  on Sales.ClientId = Clients.ClientId |

# Описание программного продукта

Для создания программы я использовал платформу Web Api ASP.NET. В среде разработки Microsoft Visual Studio. В платформе Web Api ASP.NET я сделал HTTP запросы в базу данных, с помощью swagger.json. Были сделаны различные запросы к БД по каждой таблице. Такие как: Вывод всех записей, поиск по id, поиск по наименованию, добавление данных, обновление данных, удаление данных. Это предоставляет удобное использование базы данных.

**Листинги программы**

Context представлен на листинге 4

Листинг 4. Context

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  using System.IO;  using YPYuldashevOgaySP.Models;  namespace YPYuldashevOgaySP.Contexts  {  namespace YPYuldashevOgaySP.Contexts  {  public class Context : DbContext  {  public DbSet<Client> Clients { get; set; }  public DbSet<Fuel> Fuels { get; set; }  public DbSet<Sale> Sales { get; set; }  public DbSet<Supplier> Suppliers { get; set; }  public Context(DbContextOptions<Context> options)  : base(options)  {  Database.EnsureCreated();  }  }  }  } |

**Controllers:**

Clients представлен на листинге 5

Листинге 5. Clients

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using YPYuldashevOgaySP.Contexts.YPYuldashevOgaySP.Contexts;  using YPYuldashevOgaySP.Models;  namespace YPYuldashevOgaySP.Controllers  {  [ApiController]  [Route("API/[controller]")]  public class ClientsController : ControllerBase  {  private IDbContextFactory<Context> \_contextFactory;  public ClientsController(IDbContextFactory<Context> contextFactory)  {  \_contextFactory = contextFactory;  }  /// <summary>  /// Все записи  /// </summary>  /// <returns></returns>  [HttpGet]  public async Task<IEnumerable<Client>> GetAll()  {  Context context = \_contextFactory.CreateDbContext();  return await context.Clients.ToListAsync();  }  /// <summary>  /// Добавление данных  /// </summary>  /// <returns></returns>  [HttpPost]  public async Task Create([FromBody] Client clients)  {  Context context = \_contextFactory.CreateDbContext();  await context.Clients.AddAsync(clients);  await context.SaveChangesAsync();  }  /// <summary>  /// Удаление клиентов  /// </summary>  /// <returns></returns>    [HttpDelete("{id}")]  public async Task Delete(int id)  {  Context context = \_contextFactory.CreateDbContext();  Client ClientForDelete = await context.Clients.FindAsync(id);  context.Clients.Remove(ClientForDelete);  await context.SaveChangesAsync();  }  /// <summary>  /// Поиск по id  /// </summary>  /// <returns></returns>  [HttpGet("{id}")]  public async Task<Client> Get(int id)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Clients.FindAsync(id);  }  /// <summary>  /// Поиск по наименованию  /// </summary>  /// <returns></returns>  [HttpGet("ByFullname/{Fullname}")]  public async Task<IEnumerable<Client>> GetByFullname(string Fullname)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Clients.Where(Fuel => Fuel.Fullname == Fullname).ToListAsync();  }  /// <summary>  /// Обновление данных  /// </summary>  /// <returns></returns>    [HttpPut]  public async Task Update([FromBody] Client client)  {  Context context = \_contextFactory.CreateDbContext();  context.Clients.Update(client);  await context.SaveChangesAsync();  }  }  } |

Fuels представлен на листинге 6

Листинг 6. Fuels

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using YPYuldashevOgaySP.Contexts.YPYuldashevOgaySP.Contexts;  using YPYuldashevOgaySP.Models;  namespace YPYuldashevOgaySP.Controllers  {  [ApiController]  [Route("API/[controller]")]  public class FuelsController : ControllerBase  {  private IDbContextFactory<Context> \_contextFactory;  public FuelsController(IDbContextFactory<Context> contextFactory)  {  \_contextFactory = contextFactory;  }  /// <summary>  /// Все записи  /// </summary>  /// <returns></returns>  [HttpGet]  public async Task<IEnumerable<Fuel>> GetAll()  {  Context context = \_contextFactory.CreateDbContext();  return await context.Fuels.ToListAsync();  }  /// <summary>  /// Добавление данных  /// </summary>  /// <returns></returns>  ///  [HttpPost]  public async Task Create([FromBody] Fuel fuels)  {  Context context = \_contextFactory.CreateDbContext();  await context.Fuels.AddAsync(fuels);  await context.SaveChangesAsync();  }  /// <summary>  /// Удаление  /// </summary>  /// <returns></returns>  [HttpDelete("{id}")]  public async Task Delete(int id)  {  Context context = \_contextFactory.CreateDbContext();  Fuel fuelForDelete = await context.Fuels.FindAsync(id);  context.Fuels.Remove(fuelForDelete);  await context.SaveChangesAsync();  }  /// <summary>  /// Поиск по id  /// </summary>  /// <returns></returns>  ///  [HttpGet("{id}")]  public async Task<Fuel> Get(int id)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Fuels.FindAsync(id);  }  /// <summary>  /// Поиск по наименованию  /// </summary>  /// <returns></returns>  [HttpGet("ByFullname/{Fullname}")]  public async Task<IEnumerable<Fuel>> GetByFullname(string Fullname)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Fuels.Where(Fuel => Fuel.Fullname == Fullname).ToListAsync();  }  /// <summary>  /// Обновление данных  /// </summary>  /// <returns></returns>  [HttpPut]  public async Task Update([FromBody] Fuel fuel)  {  Context context = \_contextFactory.CreateDbContext();  context.Fuels.Update(fuel);  await context.SaveChangesAsync();  }  }  } |

Sales представлен на листинге 7

Листинг 7. Sales

|  |
| --- |
| using System.IO;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using YPYuldashevOgaySP.Contexts.YPYuldashevOgaySP.Contexts;  using YPYuldashevOgaySP.Models;  namespace YPYuldashevOgaySP.Controllers  {  [ApiController]  [Route("API/[controller]")]  public class SalesController : ControllerBase  {  private IDbContextFactory<Context> \_contextFactory;  public SalesController(IDbContextFactory<Context> contextFactory)  {  \_contextFactory = contextFactory;  }  /// <summary>  /// Все записи  /// </summary>  /// <returns></returns>  [HttpGet]  public async Task<IEnumerable<Sale>> GetAll()  {  Context context = \_contextFactory.CreateDbContext();  return await context.Sales.ToListAsync();  }  /// <summary>  /// Добавление данных  /// </summary>  /// <returns></returns>  [HttpPost]  public async Task Create([FromBody] Sale sales)  {  Context context = \_contextFactory.CreateDbContext();  await context.Sales.AddAsync(sales);  await context.SaveChangesAsync();  }  /// <summary>  /// Удаление  /// </summary>  /// <returns></returns>  [HttpDelete("{id}")]  public async Task Delete(int id)  {  Context context = \_contextFactory.CreateDbContext();  Sale saleForDelete = await context.Sales.FindAsync(id);  context.Sales.Remove(saleForDelete);  await context.SaveChangesAsync();  }  /// <summary>  /// Поиск по id  /// </summary>  /// <returns></returns>    [HttpGet("{id}")]  public async Task<Sale> Get(int id)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Sales.FindAsync(id);  }  /// <summary>  /// Поиск по наименованию  /// </summary>  /// <returns></returns>  [HttpGet("ByVolume/{Volume}")]  public async Task<IEnumerable<Sale>> GetByFullname(string Volume)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Sales.Where(Sale => Sale.Volume == Volume).ToListAsync();  }  /// <summary>  /// Поиск по дате  /// </summary>  /// <returns></returns>  [HttpGet("ByDate/{date}")]  public async Task<IEnumerable<Sale>> Get(DateTime date)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Sales.Where(DateOfSale =>  DateOfSale.DateOfSale.Date == date).ToListAsync();  }  /// <summary>  /// Обновление данных  /// </summary>  /// <returns></returns>  [HttpPut]  public async Task Update([FromBody] Sale sale)  {  Context context = \_contextFactory.CreateDbContext();  context.Sales.Update(sale);  await context.SaveChangesAsync();  }  }  } |

Suppliers представлен на листинге 8

Листинг 8. Suppliers

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using YPYuldashevOgaySP.Contexts.YPYuldashevOgaySP.Contexts;  using YPYuldashevOgaySP.Models;  namespace YPYuldashevOgaySP.Controllers  {  [ApiController]  [Route("API/[controller]")]  public class SuppliersController : ControllerBase  {  private IDbContextFactory<Context> \_contextFactory;  public SuppliersController(IDbContextFactory<Context> contextFactory)  {  \_contextFactory = contextFactory;  }  /// <summary>  /// Все записи  /// </summary>  /// <returns></returns>  [HttpGet]  public async Task<IEnumerable<Supplier>> GetAll()  {  Context context = \_contextFactory.CreateDbContext();  return await context.Suppliers.ToListAsync();  }  /// <summary>  /// Добавление данных  /// </summary>  /// <returns></returns>    [HttpPost]  public async Task Create([FromBody] Supplier suppliers)  {  Context context = \_contextFactory.CreateDbContext();  await context.Suppliers.AddAsync(suppliers);  await context.SaveChangesAsync();  }  /// <summary>  /// Удаление  /// </summary>  /// <returns></returns>    [HttpDelete("{id}")]  public async Task Delete(int id)  {  Context context = \_contextFactory.CreateDbContext();  Supplier supplierForDelete = await context.Suppliers.FindAsync(id);  context.Suppliers.Remove(supplierForDelete);  await context.SaveChangesAsync();  }  /// <summary>  /// Поиск по id  /// </summary>  /// <returns></returns>    [HttpGet("{id}")]  public async Task<Supplier> Get(int id)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Suppliers.FindAsync(id);  }  /// <summary>  /// Поиск по наименованию  /// </summary>  /// <returns></returns>  [HttpGet("ByFullname/{Fullname}")]  public async Task<IEnumerable<Supplier>> GetByFullname(string Fullname)  {  Context context = \_contextFactory.CreateDbContext();  return await context.Suppliers.Where(Supplier => Supplier.Fullname == Fullname).ToListAsync();  }  /// <summary>  /// Обновление данных  /// </summary>  /// <returns></returns>  [HttpPut]  public async Task Update([FromBody] Supplier supplier)  {  Context context = \_contextFactory.CreateDbContext();  context.Suppliers.Update(supplier);  await context.SaveChangesAsync();  }  }  } |

**Models**

Модель Clients представлен на листинге

Листинг 9. Clients

|  |
| --- |
| public class Client  {  public int ClientId { get; set; }  public string Fullname { get; set; }  public string Phone { get; set; }  } |

Модель Fuels представлен на листинге 10

Листинг 10. Fuels

|  |
| --- |
| public class Fuel  {  public int FuelId { get; set; }  public string Fullname { get; set;}  public string TypeOfFuel { get; set;}  public string Price { get; set;}  public int SupplierId { get; set; }  } |

Модель Sales представлен на листинге 11

Листинг 11. Sales

|  |
| --- |
| public class Sale  {  public int SaleId { get; set; }  public string Volume { get; set;}  public string Amount { get; set;}  public DateTime DateOfSale { get; set; }  public string CheckNumber { get; set; }  public int ClientId { get; set; }  public int FuelId { get; set; }  } |

Модель Suppliers представлен на листинге 12

Листинг 12. Suppliers

|  |
| --- |
| public class Supplier  {  public int SupplierId { get; set; }  public string Fullname { get; set;}  public string Phone { get; set;}  public string LegalAddress { get; set;}  } |

Program представлен на листинге 13

Листинг 13. Program

|  |
| --- |
| using System.Reflection;  using Microsoft.EntityFrameworkCore;  using YPYuldashevOgaySP.Contexts.YPYuldashevOgaySP.Contexts;  var builder = WebApplication.CreateBuilder(args);  // Add services to the container.  builder.Services.AddControllers();  // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen(options =>  {  var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";  var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);  options.IncludeXmlComments(xmlPath);  });  builder.Services.AddOpenApiDocument();  builder.Services.AddDbContextFactory<Context>(options =>  options.UseSqlServer("Server=DESKTOP-LDJ4019\\SQLEXPRESS;Database=Autorefueling;Trusted\_Connection=True;TrustServerCertificate=True;"));  var app = builder.Build();  // Configure the HTTP request pipeline.  if (app.Environment.IsDevelopment())  {  app.UseSwagger();  app.UseSwaggerUI();  }  app.UseCors();  app.MapControllers();  app.Run(); |

**Демонстрация работы приложения**

Всезапросы представлены на рисунке 3.1

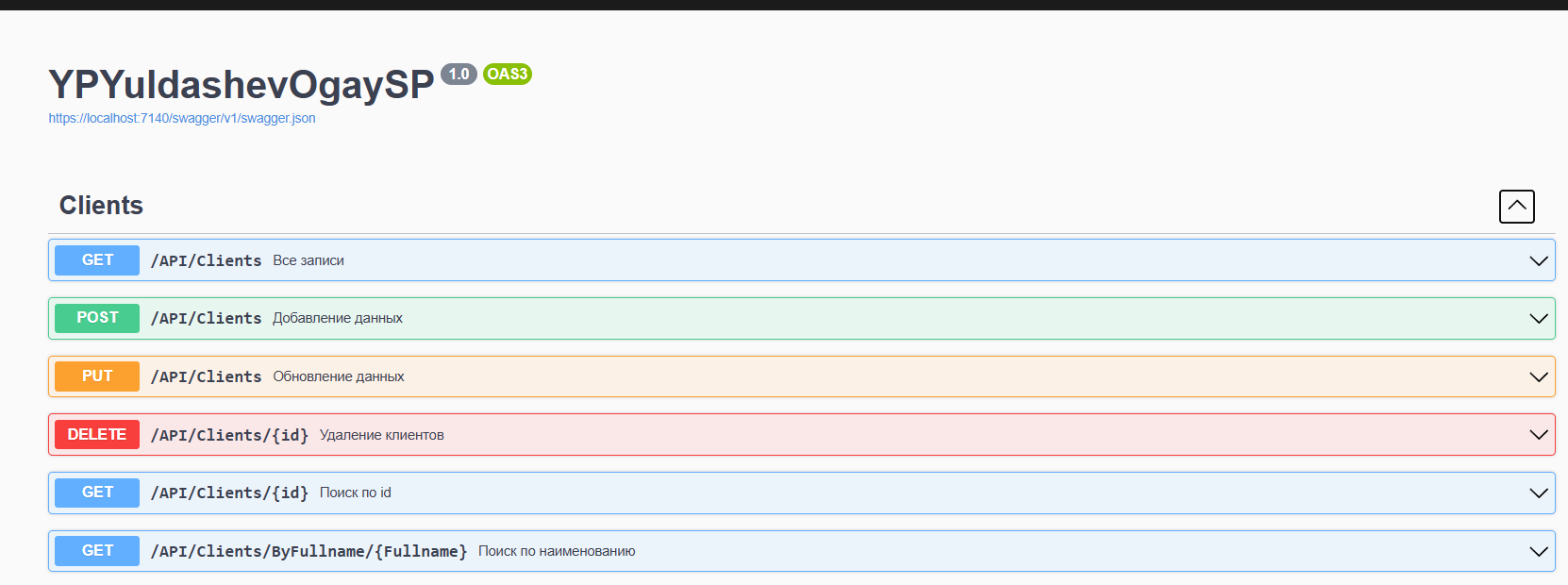
****

Рисунок 3.1 – Все запросы.

Запрос всех записей представлен на рисунке 3.2

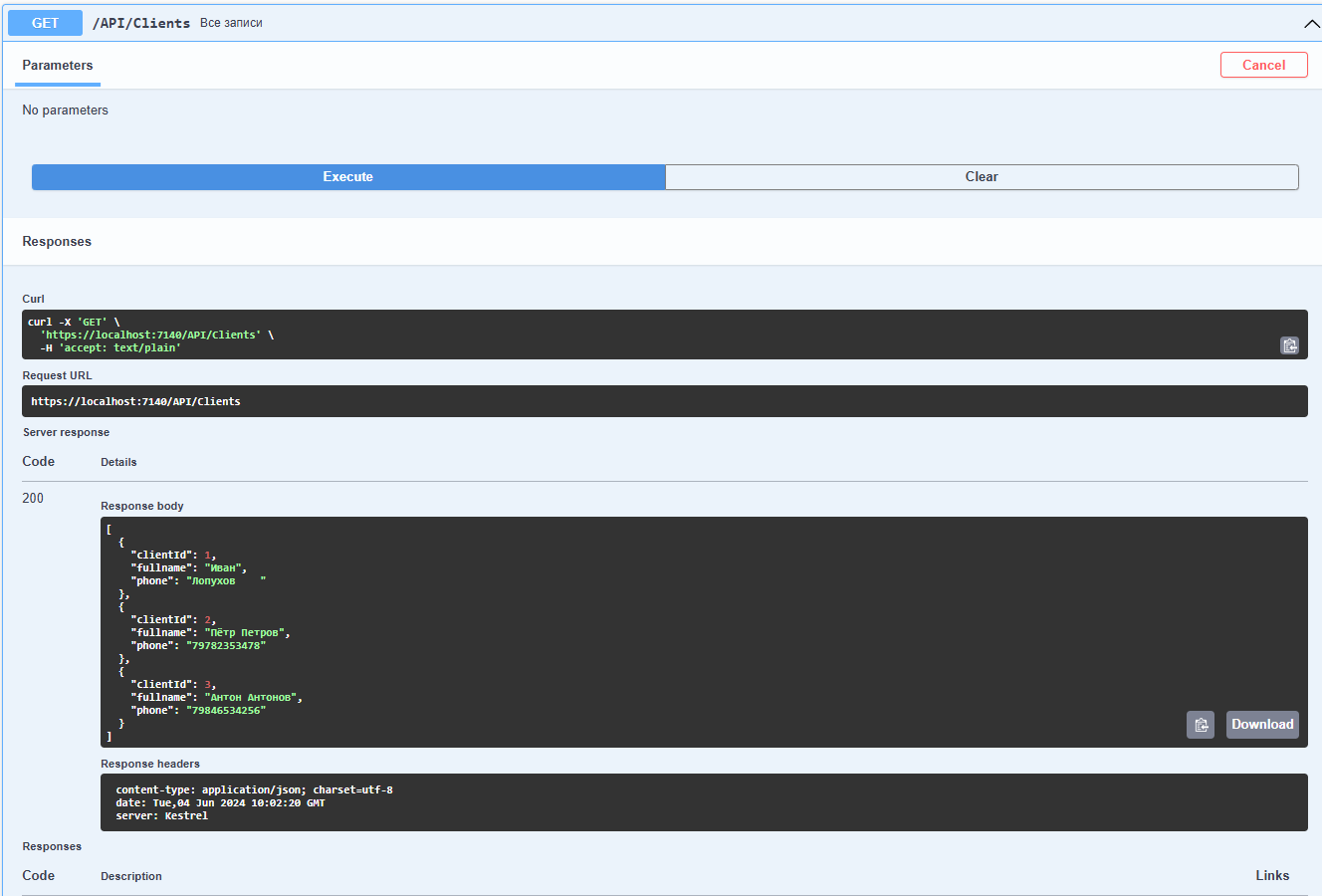


Рисунок 3.2 – Запрос всех записей.

Добавление данных представлены на рисунке 3.3

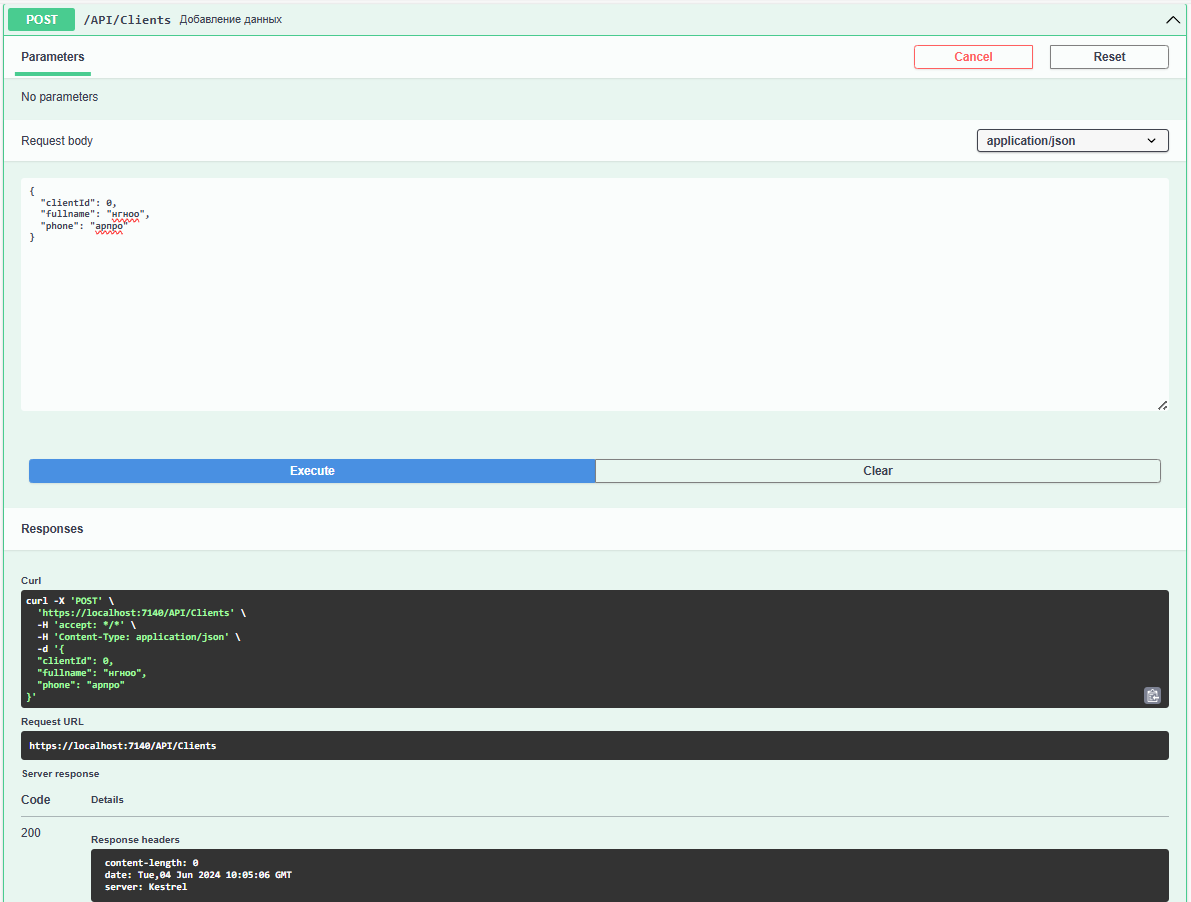


Рисунок 3.3 – Добавление данных.

Удаление данных представлены на рисунке 3.4

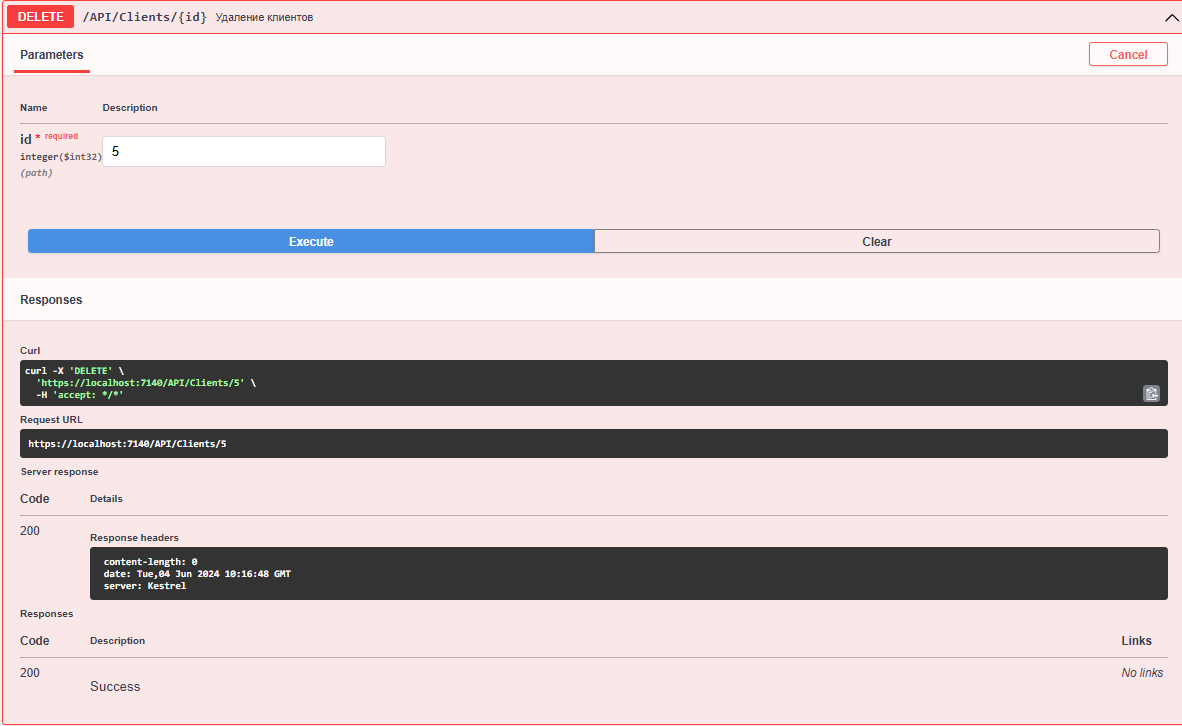


Рисунок 3.4 – Удаление данных.

Обновление данных представлен на рисунке 3.5

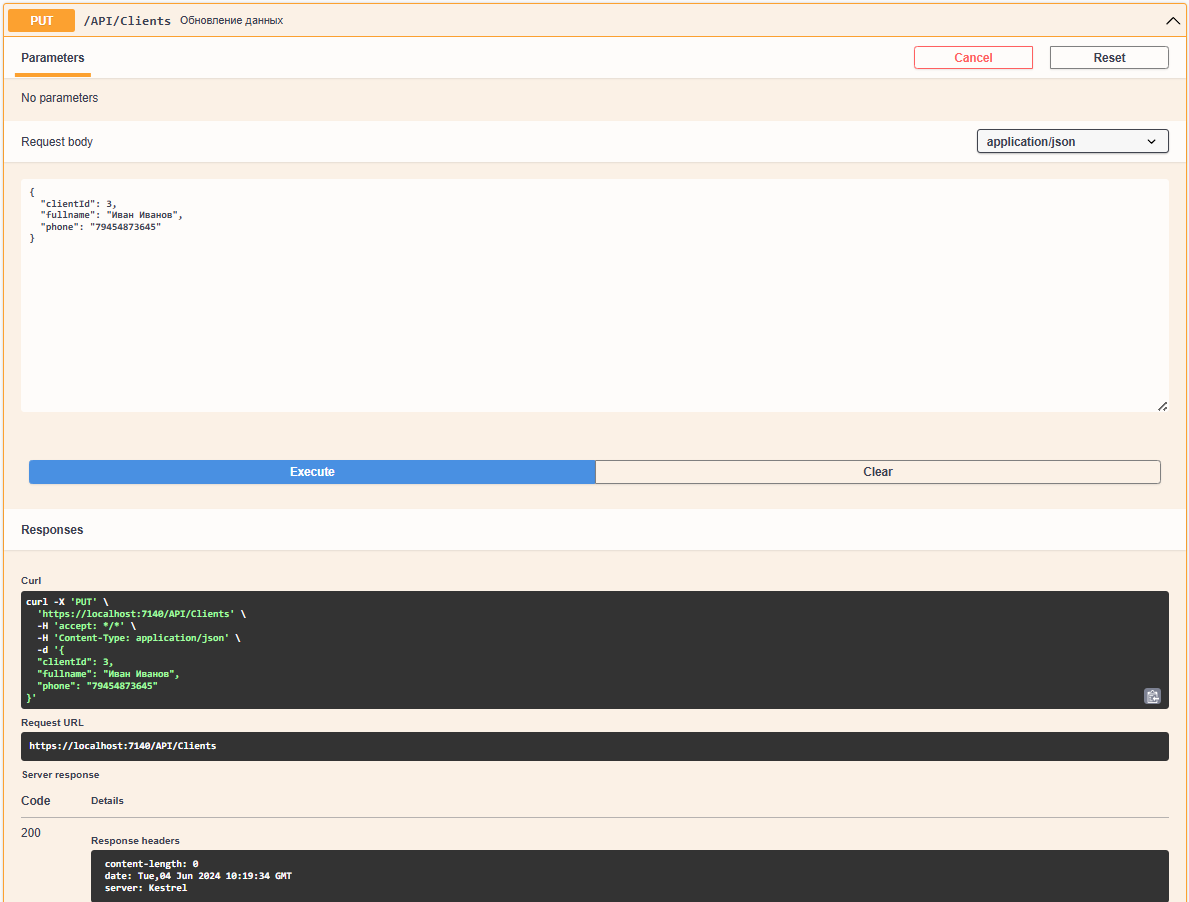


Рисунок 3.5 – Обновление данных.

Поиск данных по id представлен на рисунке 3.6

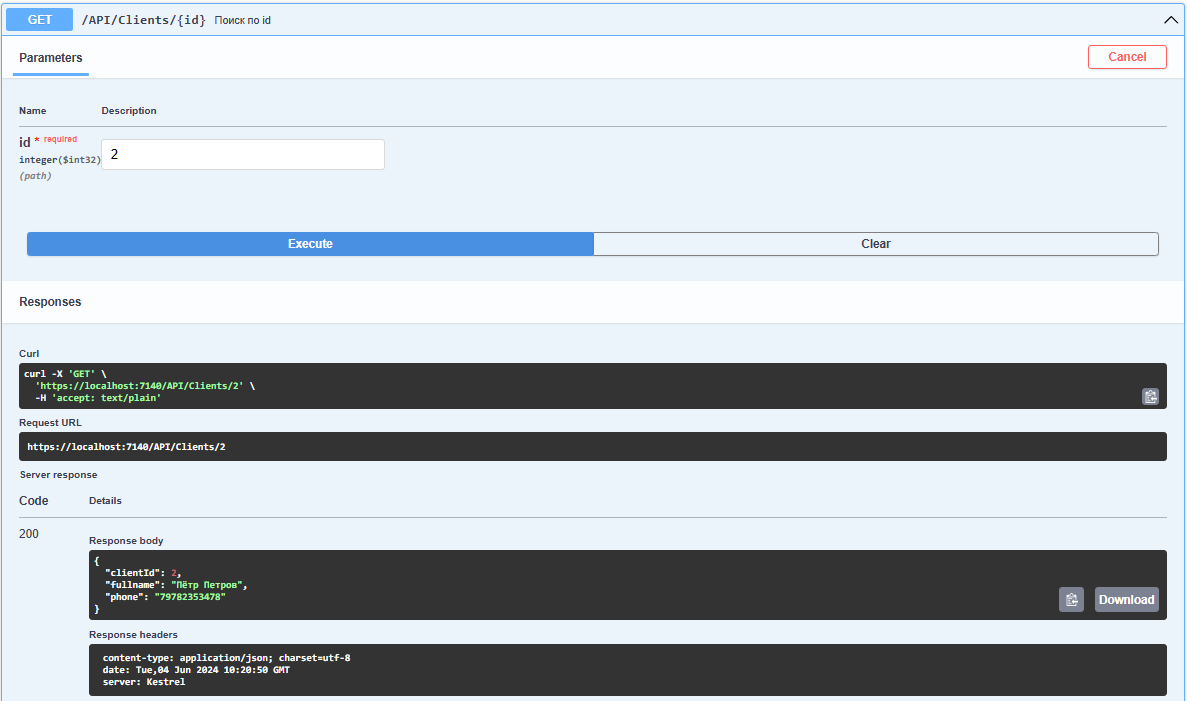


Рисунок 3.6 – Поиск данных по id.

Поиск данных по наименованию представлен на рисунке 3.7

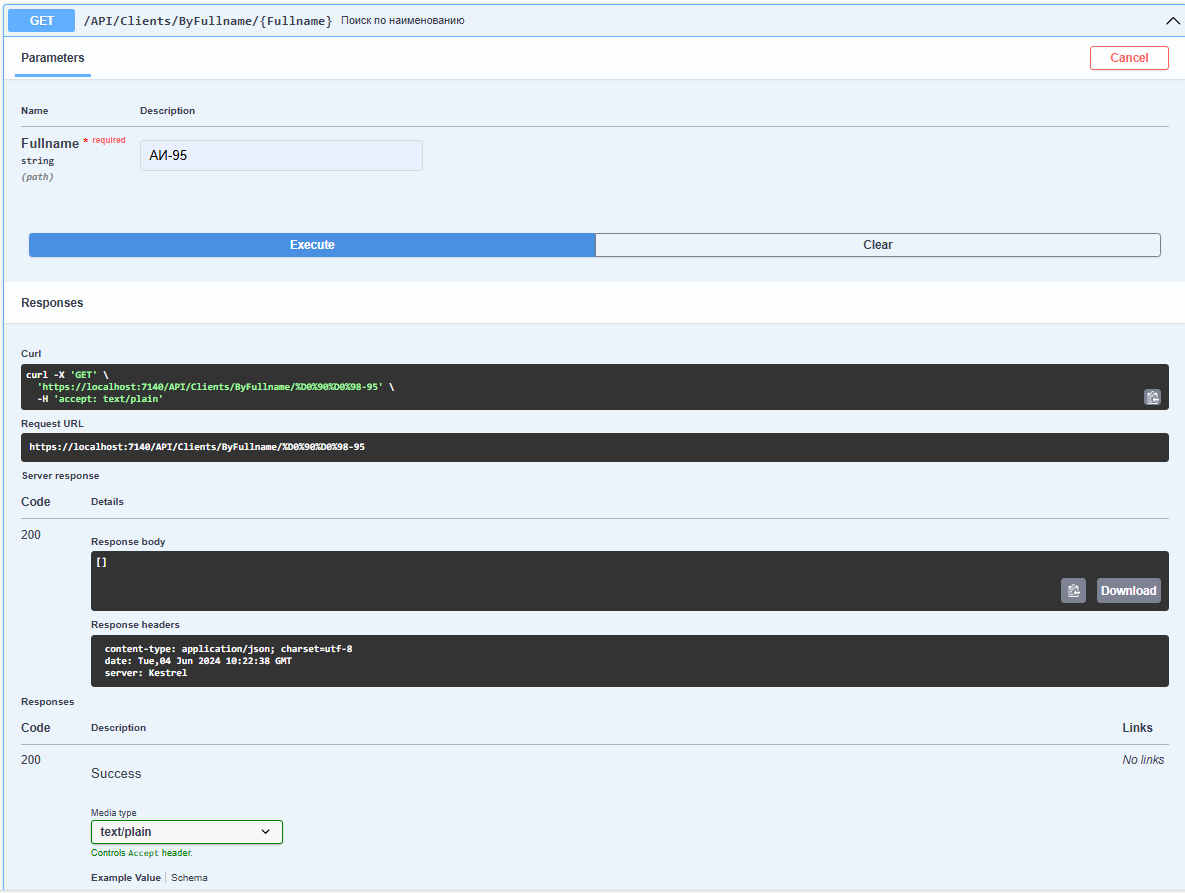


Рисунок 3.7 – Поиск данных по наименованию.

# Заключение

В ходе исследования были рассмотрены и проанализированы основные аспекты работы информационной системы заправки, а также предложены пути оптимизации процессов управления, учёта и контроля с использованием современных информационных технологий.

Для достижения поставленной цели были решены следующие задачи:

Проведён анализ существующих методов и подходов к управлению АЗС.

Разработана модель информационной системы, учитывающая особенности работы конкретной АЗС.

Внедрена разработанная система на практике и оценена её эффективность.

Исследование показало, что использование API, WPF и SQL в информационной системе заправки позволяет значительно повысить эффективность работы АЗС и удовлетворённость клиентов. Применение этих технологий обеспечивает автоматизацию процессов управления и учёта, упрощает взаимодействие с поставщиками топлива и клиентами, а также позволяет оперативно реагировать на изменения в работе станции.

Результаты исследования могут быть использованы для разработки и внедрения аналогичных информационных систем на других АЗС, а также для дальнейшего развития и совершенствования уже существующих систем.

Таким образом, можно сделать вывод, что применение современных информационных технологий в управлении АЗС является перспективным направлением для повышения эффективности работы станций и улучшения качества обслуживания клиентов.